PDO_MySQLi class

PDO_MySQLi is a PHP class to **connect to MySQL and perform Very Easy and Safe SQL queries, using any of PDO, or MySQLi** extensions.
By default, the class uses PDO, if the PHP server not support PDO, the class will use MySQLi. But if you want, you can easily change the class settings so to use MySQLi as default option.
The class uses prepare() and execute() , the data are automatically filtered by PHP and it is added Safely in the MySQL database.
The SQL queries can contain named or question mark placeholders, and the values associated to placeholders are passed separately into an array to the class method that executes the SQL query (the sqlExecute() method).

- Bellow it is the code of the class, you can copy it. And see /test the examples to learn how to use this class.

PDO_MySQLi Properties and Method

- **$conn** represents the object instance of PDO_MySQLi class.

- **$conn->sqlExecute($sql, $values)** - this method executes the SQL query (passed in $sql). The $values parameter is optional, it is an Array with values for placeholders added in SQL query.
  - In case of SELECT or SHOW queries, the method returns an array with selected rows. For other type of queries returns True. In case of error, returns False.
- **$conn->affected_rows** - contains the number of affected rows from last Insert, Update, Delete query
- **$conn->num_rows** - number of rows from last Select /Show query.
- **$conn->num_cols** - number of fields in result, from most recent Select /Show query.
- **$conn->last_insertid** - stores the last ID into an AUTO_INCREMENT column, added by last Insert query.
- **$conn->error** - contains a string with error message in case of error, otherwise, it is False.

PDO_MySQLi class code

• This class is Free, you can use, modify, and publish it freely.

```
<?php
define('USEMOD', 'pdo');     // sets default connection method: 'pdo', or 'mysqli'

// class to connect to MySQL and perform SQL queries
// From -  http://coursesweb.net/php-mysql/
class PDO_MySQLi {
  protected $usemod = '';      // 'pdo', or 'mysqli'

  static protected $conn = false;            // stores the connection to mysql
  protected $conn_data = array();            // to store data for connecting to database
  public $affected_rows = 0;        // number of affected rows for Insert, Update, Delete
  public $num_rows = 0;             // number of rows from Select /Show results
  public $num_cols = 0;             // number of columns from Select /Show results
  public $last_insertid;            // stores the last ID in an AUTO_INCREMENT column,
after Insert query
  public $error = false;          // to store and check for errors

  function __construct($conn_data) {
    $this->conn_data = $conn_data;       // stores connection data to MySQL database
  }

  // to set the connection to mysql, with PDO, or MySQLi
  protected function setConn($conn_data) {
    // sets the connection method, check if can use pdo or mysqli
    if(USEMOD == 'pdo') {
      if(extension_loaded('PDO') === true) $this->usemod = 'pdo';
      else if(extension_loaded('mysqli') === true) $this->usemod = 'mysqli';
    }
    else if(USEMOD == 'mysqli') {
      if(extension_loaded('mysqli') === true) $this->usemod = 'mysqli';
```

```
      else if(extension_loaded('PDO') === true) $this->usemod = 'pdo';
    }

    if($this->usemod == 'pdo') $this->connPDO($conn_data);
    else if($this->usemod == 'mysqli') $this->connMySQLi($conn_data);
    else $this->setSqlError('Unable to use PDO or MySQLi');
  }

  // for connecting to mysql with PDO
  protected function connPDO($conn_data) {
    try {
      // Connect and create the PDO object
      self::$conn = new PDO("mysql:host=".$conn_data['host']."; dbname=".
$conn_data['bdname'], $conn_data['user'], $conn_data['pass']);

      // Sets to handle the errors in the ERRMODE_EXCEPTION mode
      self::$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

      // Sets transfer with encoding UTF-8
      self::$conn->exec('SET
character_set_client="utf8",character_set_connection="utf8",character_set_results="utf8";
');
    }
    catch(PDOException $e) {
      $this->setSqlError($e->getMessage());
    }
  }

  // method that create the connection to mysql with MySQLi
  protected function connMySQLi($conn_data) {
    // if the connection is successfully established
    if(self::$conn = new mysqli($conn_data['host'], $conn_data['user'],
$conn_data['pass'], $conn_data['bdname'])) {
      self::$conn->query('SET
character_set_client="utf8",character_set_connection="utf8",character_set_results="utf8";
');
    }
    else if (mysqli_connect_errno()) $this->setSqlError('MySQL connection failed: '.
mysqli_connect_error());
  }

  // Performs SQL queries
  // $sql - SQL query with prepared statement
  // $param - array of values for SQL query
  public function sqlExecute($sql, $param=array()) {
    if(self::$conn === false || self::$conn === NULL) $this->setConn($this->conn_data);
// sets the connection to mysql
    $re = true;           // the value to be returned

    // if there is a connection set ($conn property not false)
    if(self::$conn !== false) {
      // gets the first word in $sql, to determine whenb SELECT query
      $ar_mode = explode(' ', trim($sql), 2);
      $mode = strtolower($ar_mode[0]);

      // replace ":for_value" with "?" (for MySQLi)
      if($this->usemod == 'mysqli') $sql = preg_replace('/:[^,|"|\'|;|\)\} ]*/i','?',
$sql);

      $sqlre = self::$conn->prepare($sql);    // prepares statement

      // if successfully prepared
      if(is_object($sqlre)) {
        // execute query
        if($this->usemod == 'pdo') {
          try { $sqlre->execute($param); }
          catch(PDOException $e) { $this->setSqlError($e->getMessage()); }
        }
```

```
        else if($this->usemod == 'mysqli') {
          // if values in $param, sets to use "bind_param" before execute()
          if(count($param) > 0) {
            // stores in $args[0] the type of the value of each value in $param, the rest
of items in $args are the values
            $args = array('');
            foreach($param AS $k=>$v) {
              if(is_int($v)) $args[0] .= 'i';
              else if(is_double($v)) $args[0] .= 'd';
              else $args[0] .= 's';
              $args[] = &$param[$k];
            }

            // binds the values with their types in prepared statement
            call_user_func_array(array($sqlre,'bind_param'), $args);
          }

          if(!$sqlre->execute()) {
            if(isset(self::$conn->error_list[0]['error'])) $this->setSqlError(self::
$conn->error_list[0]['error']);
            else $this->setSqlError('Unable to execute the SQL query, check if values are
passed to sqlExecute()');
          }
        }
      }
      else {
        if(isset(self::$conn->error_list[0]['error'])) $this->setSqlError(self::$conn-
>error_list[0]['error']);
        else $this->setSqlError('Unable to prepare the SQL query, check if SQL query data
are correctly');
      }

      // if no error
      if($this->error === false) {
        // if $mode is 'select' or 'show', gets the result_set to return
        if($mode == 'select' || $mode == 'show') {
          $re = ($this->usemod == 'pdo') ? $this->getSelectPDO($sqlre) : $this-
>getSelectMySQLi($sqlre);    // gets select results

          $this->num_rows = count($re);                    // number of returned rows
          if(isset($re[0])) $this->num_cols = count($re[0]) / 2;           // number of
returned columns
        }
        else $this->affected_rows = ($this->usemod == 'pdo') ? $sqlre->rowCount() :
$sqlre->affected_rows;      // affected rows for Insert, Update, Delete

        // if Insert query, stores the last insert ID
        if($mode == 'insert') $this->last_insertid = ($this->usemod == 'pdo') ? self::
$conn->lastInsertId() : self::$conn->insert_id;
      }
    }

    // sets to return false in case of error
    if($this->error !== false) $re = false;
    return $re;
  }

  // gets and returns Select results performed with PDO
  // receives the object created with prepare() statement
  protected function getSelectPDO($sqlre) {
    $re = array();
    // if fetch() returns at least one row (not false), adds the rows in $re for return
(numerical, and associative)
    if($row = $sqlre->fetch()){
      do {
        // check each column if it has numeric value, to convert it from "string"
        foreach($row AS $k=>$v) {
          if(is_numeric($v)) $row[$k] = $v + 0;
```

```
        }
        $re[] = $row;
      }
      while($row = $sqlre->fetch());
    }

    return $re;
  }

  // gets and returns Select results performed with MySQLi
  // receives the object created with prepare() statement
  protected function getSelectMySQLi($sqlre) {
    $meta = $sqlre->result_metadata();
    $re = array(); $parameters = array();

    // gets column names, to be passed as parameters to bind_result()
    while ($field = $meta->fetch_field()) {
      $parameters[] = &$row[$field->name];
    }

    // accesses $sqlre->bind_result with arguments stored in $parameters
    call_user_func_array(array($sqlre, 'bind_result'), $parameters);

    // gets array with results (numerical ($x1), and associative ($x2))
    while($sqlre->fetch()) {
      $i=0;
      foreach($row as $k => $v) {
        $x1[$i] = $v;
        $x2[$k] = $v;
        $i++;
      }
      $re[] = array_merge($x1, $x2);
    }

    return $re;
  }

  // set sql error in $error
  protected function setSqlError($err) {
    $this->error = '<h4>Error: '. $err .'</h4>';
  }
}
```

For SQL queries that affects data in MySQL database, it is indicated to use placeholder names (or question marks) in the SQL statement, and pass the associated values separately into an array. Data will be added Safely in MySQL database.

PDO_MySQLi usage

1. Copy the *PDO_MySQLi class* on your server (for example into a PHP file named "class.pdo_mysqli.php").
2. In the PHP file in which you want to use the class, include it, and create an object instance of PDO_MySQLi class, passing as argument an array with your data for connecting to MySQL database.

```
<?php
// Array with data for connecting to mysql database
$mysql = array(
 'host' => 'localhost',
 'user' => 'root',
 'pass' => 'password',
 'bdname' => 'db_name'
);

include('class.pdo_mysqli.php');     // includes the php file with PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);
```

3. Define the SQL query (optional, array with values for placeholders in SQL), and call the sqlExecute() method, passing the SQL query string, and optional the array with values. You can use named or question mark placeholders in the SQL statement (see also in the examples bellow, or from downloaded archive).

```
// Select with named placeholder
$sql = "SELECT * FROM table_name WHERE column = :val";
$values = array('val'=>'value');
$rows = $conn->sqlExecute($sql, $values);

// OR, Select without placeholders
$sql = "SELECT * FROM table_name WHERE id = 8";
$rows = $conn->sqlExecute($sql);

// traverses the array with rows data
foreach($rows AS $row) {
  echo $row['column'];
}
```

• To change the default connection from PDO to MySQLi, replace "pdo" with "mysqli" value in the USEMOD constant, defined in the file with PDO_MySQLi class code (line 2).

- When it is performed Select or Show SQL query, the sqlExecute() method returns an array with selected rows. The array contains rows data associated to column names, and also indexed numerically (index from 0) associated to columns order. The array can be traversed using foreach() or for() statement.
For various type of SQL queries, see the code and comments in the examples presented bellow.

Examples usage of PDO_MySQLi class

Ten examples with usage of PDO_MySQLi class.
First, it must be created the array with data for connecting to MySQL database, and include the file with PDO_MySQLi class, like in this example.

```
<?php
// Array with data for connecting to mysql database
$mysql = array(
 'host' => 'localhost',
 'user' => 'root',
 'pass' => 'password',
 'bdname' => 'db_name'
);

include('class.pdo_mysqli.php');     // includes the file with PDO_MySQLi class
```

Now, we create the object instance of PDO_MySQLi class, and perform some SQL queries: CREATE TABLE, INSERT, UPDATE, SELECT, DELETE, and SHOW (see the comments in code).

1. Create table.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// CREATES TABLE
$sql = "CREATE TABLE `testclass` (`id` INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, `url`
VARCHAR(255), `title` VARCHAR(125), `dt` INT(11) NOT NULL) CHARACTER SET utf8 COLLATE
utf8_general_ci";

// executes the SQL query, passing only the string with SQL query
$resql = $conn->sqlExecute($sql);
```

```
// check if the SQL query succesfully performed, else outputs the error
if($resql) echo 'The table "testclass" succesfully created';
else echo $conn->error;
```

Results:
The table "testclass" succesfully created.

2. Insert one row, using placeholder names in SQL statement.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// INSERT one row, with corresponding placeholder names in the SQL statement, and
array with values for associated names
$sql = "INSERT INTO `testclass` (`url`, `title`, `dt`) VALUES (:url, :title, :dt)";
$vals = array('url'=>'http://coursesweb.net/', 'title'=>'Web Development Courses',
'dt'=>time());

// executes the SQL query, passing the SQL and the array with values
$resql = $conn->sqlExecute($sql, $vals);

$last_id = $conn->last_insertid;     // gets last inserted Auto-Increment ID

// check if the SQL query succesfully performed, and displays the number of affected
(inserted) rows, and last ID
if($resql) echo 'Inserted succesfully '. $conn->affected_rows .' row. Last inserted
ID: '. $last_id;
else echo $conn->error;
```

Results:
Inserted succesfully 1 row. Last inserted ID: 1

3. Insert two rows in the same SQL query.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// INSERT two rows in the same query
// with corresponding placeholder names in the SQL statement, and array with values
for corresponding names
$sql = "INSERT INTO `testclass` (`url`, `title`, `dt`) VALUES (:url1, :title1, :dt1),
(:url2, :title2, :dt2)";
$vals = array(
  'url1'=>'http://coursesweb.net/php-mysql/', 'title1'=>'PHP-MySQL free course',
'dt1'=>time(),
  'url2'=>'http://coursesweb.net/javascript/', 'title2'=>'JavaScript & jQuery
Course', 'dt2'=>time()
);

// executes the SQL query, passing the SQL query and the array with values
$resql = $conn->sqlExecute($sql, $vals);

/*
 when there are inserted multiple rows in the same query, the last insert id in an
auto-increment column will be the id number of the first inserted row. In this case,
to get the real last insert id, add to $last_insertid the number of inserted rows
```

```
less one.
*/
$last_id = $conn->last_insertid + (2 - 1);

// check if the SQL query succesfully performed
// displays the number of affected (inserted) rows, and the last auto_increment ID
if($resql) echo 'Inserted succesfully '. $conn->affected_rows .' rows. Last Auto-
Increment ID: '. $last_id;
else echo $conn->error;
```

Outputs:
Inserted succesfully 2 rows. Last Auto-Increment ID: 3

4. Update, using corresponding placeholder names

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// UPDATE with corresponding placeholder names in the SQL statement, and the values
into an associative array
$sql = "UPDATE `testclass` SET `title` = :title WHERE `url` = :url";
$vals = array('title'=>'JavaScript & jQuery',
'url'=>'http://coursesweb.net/javascript/');

// executes the SQL query, passing the SQL query and the array with values
$resql = $conn->sqlExecute($sql, $vals);

// check if the SQL query succesfully performed, displays the number of affected rows
if($resql) echo 'Updated succesfully '. $conn->affected_rows .' row';
else echo $conn->error;
```

Outputs:
Updated succesfully 1 row

5. Update, using placeholder question marks in the SQL statement.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// UPDATE with placeholder question marks in the SQL statement,
// and values into an numerical array (in the same order as question marks), numerical
index starting from 0
$sql = "UPDATE `testclass` SET `url` = ?, `title` = ? WHERE `id` = ?";
$vals = array('http://www.marplo.net/', 'Free Courses', 3);

// executes the SQL query, passing the SQL query and the array with values
$resql = $conn->sqlExecute($sql, $vals);

// check if the SQL query succesfully performed, displays the number of affected rows
if($resql) echo 'Updated succesfully '. $conn->affected_rows .' row';
else echo $conn->error;
```

Results:
Updated succesfully 1 row

6. Simple Select, without placeholders. Using foreach() to traverse the selected rows.

```
<?php
```

```
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// simple SELECT, without placeholders
$sql = "SELECT * FROM `testclass` WHERE `id` > 1";

// executes the SQL query (passing only the SQL query), and gets the selected rows
$rows = $conn->sqlExecute($sql);

$nr_rows = $conn->num_rows;          // number of selected rows
$nr_cols = $conn->num_cols;          // number of selected columns

// if there are returned rows, traverses the array with rows data, using foreach()
if($nr_rows > 0) {
  echo 'Number of selected rows: '. $nr_rows .' / Number of columns: '. $nr_cols;

  foreach($rows AS $row) {
    echo '<br/>ID = '. $row['id'] .' / Title = '. $row['title'];
  }
}
else {
  if($conn->error) echo $conn->error;      // if error, outputs it
  echo '0 selected rows';
}
```

Results:

Number of selected rows: 2 / Number of columns: 4
ID = 2 / Title = PHP-MySQL free course
ID = 3 / Title = Free Courses

7. Select, with named placeholders.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

/*
  for values used in LIKE statement, add the "?" or "%" characters in the value. Use like
this
  $val = '%'. $val .'?';
*/

// SELECT with named placeholders and values for placeholders added into an associative
array
$sql = "SELECT * FROM `testclass` WHERE `title` LIKE :title ORDER BY `id`";
$vals = array('title'=> '%Course%');

// executes the SQL query (passing the SQL query, and array with values), and gets the
selected rows
$rows = $conn->sqlExecute($sql, $vals);

$nr_rows = $conn->num_rows;          // number of selected rows
$nr_cols = $conn->num_cols;          // number of selected columns

// if there are returned rows, traverses the array with rows data, using foreach()
if($nr_rows > 0) {
  echo 'Number of selected rows: '. $nr_rows .' / Number of columns: '. $nr_cols;

  // outputs data using index number of column order (index starting from 0)
  foreach($rows AS $row) {
    echo '<br/>Col1 = '. $row[0] .' / Col2 = '. $row[1] .' / Col3 = '. $row[2] .' / Col4
```

```
= '. $row[3];
  }
}
else {
  if($conn->error) echo $conn->error;      // if error, outputs it
  echo '0 selected rows';
}
```

Outputs:

Number of selected rows: 3 / Number of columns: 4
Col1 = 1 / Col2 = http://coursesweb.net/ / Col3 = Web Development Courses / Col4 = 1377928573
Col1 = 2 / Col2 = http://coursesweb.net/php-mysql/ / Col3 = PHP-MySQL free course / Col4 = 1377928814
Col1 = 3 / Col2 = http://www.marplo.net// Col3 = Free Courses / Col4 = 1377928814

8. Delete, using named placeholder.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// DELETE with named placeholder and associated value added into an array
$sql = "DELETE FROM `testclass` WHERE `id` < :id";
$vals = array('id'=>3);

// executes the SQL query, passing the SQL query and the array with value
$resql = $conn->sqlExecute($sql, $vals);

// check if the SQL query succesfully performed, displays the number of affected rows
if($resql) echo 'Deleted succesfully '. $conn->affected_rows .' rows';
else echo $conn->error;
```

Results:

Deleted succesfully 2 rows

9. It can be performed multiple SQL queries with the same object instance of PDO_MySQLi class. It this example there are performed an Insert and Select queries; in the Select statement it is used the Last Inserted ID returned by the previous Insert.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// Insert, with question marks placeholders, and values added into an array (in the same
order associated to question marks)
$sql = "INSERT INTO `testclass` (`url`, `title`, `dt`) VALUES (?, ?, ?)";
$vals = array('http://coursesweb.net/', 'Free Web Courses', time());

// executes the SQL query, passing the SQL query and the array with values
$resql = $conn->sqlExecute($sql, $vals);

// check if the SQL query succesfully performed
if($resql) {
  // outputs the number of affected (inserted) rows, and last inserted ID
  $last_id = $conn->last_insertid;
  echo 'Inserted succesfully '. $conn->affected_rows .' row. Last Auto-Increment ID: '.
$last_id;

  // simple SELECT (without placeholders) using the last-inserted-id
  $sql = "SELECT `id`, `title`, `url` FROM `testclass` WHERE `id` NOT IN( $last_id )";

  // executes the SQL query using the same connection object, and gets the selected rows
```

```
  $rows = $conn->sqlExecute($sql);

  $nr_rows = $conn->num_rows;          // number of selected rows

  // if there are returned rows, traverses the array with rows data, using for()
instruction
  if($nr_rows > 0) {
    for($i=0; $i<$nr_rows; $i++) {
      $row = $rows[$i];
      echo '<br/>ID = '. $row['id'] .' / URL = '. $row['url'] .' / Title = '.
$row['title'];
    }
  }
  else {
    if($conn->error) echo $conn->error;      // if error, outputs it
    echo '0 selected rows';
  }
}
else echo $conn->error;
```

Outputs:

Inserted succesfully 1 row. Last Auto-Increment ID: 4
ID = 3 / URL = http://www.marplo.net// Title = Free Courses

10. Show, gets the name and type of the columns in MySQL table.

```
<?php
// Here it is added /included the code with data for connecting to MySQL, and the
PDO_MySQLi class

// creates object with connection to MySQL
$conn = new PDO_MySQLi($mysql);

// simple SHOW, without placeholders
$sql = "SHOW COLUMNS FROM `testclass`";

// executes the SQL query (passing only the SQL query), and gets the selected rows
$rows = $conn->sqlExecute($sql);

// gets the number of rows with returned data (number of columns in table)
$nr_c = $conn->num_rows;

// if there are returned columns, traverses the array with columns data, using foreach(),
and outputs each column name and its type
if($nr_c > 0) {
  echo 'Number of columns: '. $nr_c;

  foreach($rows AS $row) {
    echo '<br/>Column = "'. $row['Field'] .'" / Type = '. $row['Type'];
  }
}
else {
  if($conn->error) echo $conn->error;      // if error, outputs it
  echo '0 selected results';
}
```

Results:

Number of columns: 4
Column = "id" / Type = int(10) unsigned
Column = "url" / Type = varchar(255)
Column = "title" / Type = varchar(125)
Column = "dt" / Type = int(11)